MYSTRAN NOTES

Updated 01/17/2023 By Harry Schaeffer

Link1 Generate Element Stiffness

Modifications to pass model definitions to NEWQ4

Add COMMON /HGS/ that be used in several downstream modules

Common /hgs/

! Allocate several arrays since they are needed every time LINK1 gets re-run (for BUCKLING and NLSTATIC). They were used in
! LINK0 but deallocated there at the end of LINK0 so that we could allocate them again here

```
      CALL ALLOCATE_MODEL_STUF ( 'SINGLE ELEMENT ARRAYS', SUBR_NAME )
      ELSE IF (NAME_IN == 'SINGLE ELEMENT ARRAYS') THEN        ! Allocate arrays for elem thermal and pressure loads

      NAME = 'AGRID'
```

Here is the brang to static processing

```
      IF ((SOL_NAME(1:7) == 'STATICS') .OR. (SOL_NAME(1:8) == 'NLSTATIC') .OR.
      &
      ((SOL_NAME(1:8) == 'BUCKLING') .AND. (LOAD_ISTEP == 1))) THEN
```

Here is the call to esp (Stiffness processor) from EPTL

```
      CALL OURTIM
      MODNAM = 'G-SET STIFFNESS MATRIX PROCESSOR             '
      WRITE(SC1,1092) LINKNO,MODNAM,HOUR,MINUTE,SEC,SFRAC
      CALL ESP
```

LINK1/EPTL/ESP

```
USE MODEL_STUF, ONLY              :  AGRID, ELDT, ELDOF, ELGP, GRID_ID, NUM_EMG_FATAL_ERRS, PLY_NUM, OELDT, KE, KED, TYPE
```

ASET is in scope

;

OPT Flags are set

```
        OPT(1) = 'N'                        ! OPT(1) is for calc of ME
        OPT(2) = 'N'                        ! OPT(2) is for calc of PTE
        OPT(3) = 'N'                        ! OPT(3) is for calc of SEi, STEi
        OPT(5) = 'N'                        ! OPT(5) is for calc of PPE
```

Followed by:

```
        OPT(4) = 'Y'                        ! OPT(4) is for calc of KE-linear
        OPT(6) = 'N'
```

Loop over all elements:

```
elems:DO I=1,NELE
```

Call EMG

LINK1/ESP/EMG

Initialize Model stuff

```
 USE MODEL_STUF, ONLY          :  AGRID, BE1, BE2, BE3, BGRID, DOFPIN, DT,
ELAS_COMP, FCONV, KE, KED, ME,                  &
                               OFFDIS, OFFSET, PEB, PEG, PEL, PPE, PRESS, PTE, SE1,
SE2, SE3, STE1, STE2, STE3,         &
                               UEB, UEG, UEL, UGG, XEB, XEL
```

! Call ELMDAT1 subr to get some of the data needed for this elem   SUBROUTINE ELMDAT1 (INT_ELEM_ID, WRITE_WARN)

! Generates small arrays of elem data, for use by subroutine EMG, one elem at a time for all elems. Arrays generated are:

!   XEB   : basic coords of grids for 1 elem

!   V VECTOR : for some 1-D elems

!   EPROP   : array of elem geometric properties

!   ISOLID   : data for 3-D elems defining options from the PSOLID Bulk Data entry

!   EMAT   : material property data

!   PIN FLAG : Pin flag data for some elems

!   OFFSETS : offsets for some elems

Model_stuff

```
 USE MODEL_STUF, ONLY            :  AGRID, BAROFF, BUSH_CID, BUSH_OCID,
BUSH_VVEC, BUSH_VVEC_OR_CID, BUSHOFF, BGRID,          &
                                 CAN_ELEM_TYPE_OFFSET, CORD, DOFPIN, EDAT, EID,
ELAS_COMP, ELDOF, ELEM_LEN_12, ELGP,       &
                                 ELMTYP, EMAT, EOFF, NUM_EMG_FATAL_ERRS, EPROP,
EPNT, ETYPE, GRID, RGRID, GRID_ID,      &
                                 INTL_MID, INTL_PID, ISOLID, MATANGLE, MATL,
MTRL_TYPE, NUM_SEi, OFFDIS, OFFDIS_O, OFFSET, &
                                 PBAR, PBEAM, PCOMP, PCOMP_PROPS, PLATEOFF,
PLATETHICK, PROD, PSHEAR, PSHEL, PSOLID,           &
                                 PUSER1, PUSERIN, RMATL, RPBAR, RPBEAM, RPBUSH,
RPELAS, RPROD, RPSHEAR, RPSHEL, RPUSER1,   &
                                 TYPE, VVEC, XEB, ZOFFS
```

In ELMDAT1: CALL GET_ELEM_AGRID_BGRID (INT_ELEM_ID, 'Y')

Bgrid is visible and correct

In emdat1: Here ie where the materia type are loaded into INTL_MID: verified for quard4

```
 Line 586: ELSE IF ((TYPE(1:5) == 'TRIA3') .OR. (TYPE(1:5) == 'QUAD4')) THEN
                                         ! For elems that are not composites do EMAT in subr
SHELL_ABD_MATRICES)
        IF (PCOMP_PROPS == 'N') THEN

        INTL_MID(1) = PSHEL(INTL_PID,2)
        IF (INTL_MID(1) /= 0) THEN
        MTRL_TYPE(1) = MATL(INTL_MID(1),2)
        ENDIF

        INTL_MID(2) = PSHEL(INTL_PID,3)
        IF (INTL_MID(2) /= 0) THEN
        MTRL_TYPE(2) = MATL(INTL_MID(2),2)
        ENDIF

        INTL_MID(3) = PSHEL(INTL_PID,4)
        IF (INTL_MID(3) /= 0) THEN
        MTRL_TYPE(3) = MATL(INTL_MID(3),2)
        ENDIF

        INTL_MID(4) = PSHEL(INTL_PID,5)
        IF (INTL_MID(4) /= 0) THEN
        MTRL_TYPE(4) = MATL(INTL_MID(4),2)
        ENDIF
```

```
          NUMMAT = 4
```

! For all but USERIN elem, call ELMDAT2 subr to get the rest of the data needed to calculate the matrices for this element.

```
          SUBROUTINE ELMDAT2 ( INT_ELEM_ID, OPT, WRITE_WARN )
```

! Generates small arrays of elem data, for use by subroutine EMG, one elem at a time for all elems. Arrays generated are:

!   DT (1 elem temperatures) and PRESS (1 element pressure load)

Model Stuf

```
USE MODEL_STUF, ONLY             :  BGRID, DT, ELGP, ETYPE, GTEMP, PDATA, PPNT,
PTYPE, PRESS, TDATA, TPNT, TYPE
```

In EMG

! Now get the individual elem routines to calc the required elem matrices: ME and/or PTE and/or (SE1, SE2, STE1,STE2)
! and/or KE).

LINK1/ESP/EMG/QDEL1

CALL QDEL1 (OPT, WRITE_WARN)
Mode_stuf

```
USE MODEL_STUF, ONLY             :  EID, ELDOF, EMG_IFE, EMG_RFE, EMAT,
ERR_SUB_NAM, EB, INTL_MID, KE,                    &
                                        MASS_PER_UNIT_AREA, NUM_EMG_FATAL_ERRS, ME,
PCOMP_LAM, PCOMP_PROPS, SHELL_B, TYPE, XEL
          USE MODEL_STUF, ONLY             :  BENSUM, SHRSUM, PHI_SQ, PSI_HAT
```

Call element routine appropriate for param quadtyp

```
IF (TYPE(1:5) == 'QUAD4' .and. quad4typ .ne. 'NEWQ4') THEN
```

For quad4typ == MIN4

```
SUBROUTINE QMEM1 ( OPT, IORD, RED_INT_SHEAR, AREA, XSD, YSD, BIG_BM )
```

Note that MS is passing argos created in QDEL1

GRID Notes from MODEL_STUF

```
! After Bulk Data is read, SEQ1 and SEQ2 are in the order in which they were encountered in the Bulk Data. They are
! initially created in subr BD_SEQP. After subr SEQ_PROC has run, they are in an order in which the grids in SEQ1 are
! in numerical order (and this is the order they are in when they are written to filename.L1B)

!
*************************************************************************************************************************
*
! Grid data
! ---------

          INTEGER(LONG), ALLOCATABLE    :: GRID(:,:)        ! Array of int data from GRID Bulk Data entries (see comments below)
          INTEGER(LONG)                 :: GRDSET3  = 0     ! Input coord system defined in field 3 of a GRDSET entries, if
present
          INTEGER(LONG)                 :: GRDSET7  = 0     ! Displ coord system defined in field 7 of a GRDSET entries, if
present
          INTEGER(LONG)                 :: GRDSET8  = 0     ! Permanent SPC's defined in field 8 of a GRDSET entries, if present
          INTEGER(LONG), ALLOCATABLE    :: GRID_ID (:)      ! Array of grid ID's in numerical order
          INTEGER(LONG), ALLOCATABLE    :: GRID_SEQ(:)      ! GRID_SEQ(i) is the sequence number for grid GRID_ID(i) (see below)

                                                            ! Array that shows the elements connected to each grid
          INTEGER(LONG), ALLOCATABLE    :: GRID_ELEM_CONN_ARRAY(:,:)

          INTEGER(LONG), ALLOCATABLE    :: INV_GRID_SEQ(:)  ! INV_GRID_SEQ(i) = internal grid ID that is sequenced i-th (see
below)

          REAL(DOUBLE) , ALLOCATABLE    :: RGRID(:,:)       ! Array of real data from GRID Bulk Data entries (see comments below)

! Each row of GRID is for one grid point and contains:

!                   (1) Grid point number         in col 1
!                   (2) Input coord system   in col 2
!                   (3) Global coord system  in col 3
!                   (4) Permanent SPC's           in col 4
!                   (5) Line break indicator in col 5 (put this many line breaks in F06 after this grid number)
!                   (6) Num of comps              in col 6 (1 for SPOINT, 6 for actual grid)

!         The array is sorted in the following order:
!                   (1) After the B.D. deck is read it is in GRID input order
!                   (2) After subr GRID_PROC it is in grid point numerical order

! Each row of RGRID is for one grid point and contains:

!         After Bulk Data has been read: the 3 coords of the grid in the coord sys defined in col 2 of array GRID for this G.P.
!         After subr GRID_PROC has run : the 3 coords of the grid in the basic (0) coord sys of the model

! The following example explains GRID_ID(I), GRID_SEQ(I) and INV_GRID_SEQ(I). The model has 5 grid points and they are sequenced
! in the order 401, 201, 501, 301, 101. Array GRID_ID has the grids in numerical order. GRID_SEQ(I) is the sequence number for
! GRID_ID(I). INV_GRID_SEQ(I) does the following: the 4 for INV_GRID_SEQ(1) means that the 4th entry in GRID_ID (grid 401) is
! sequenced as GRID_SEQ(4) or first (as stated in the example).

!                             I  GRID_ID(I)  GRID_SEQ(I)  INV_GRID_SEQ(I)
!
!                             1         101       5                  4 (i.e. the 4th entry in GRID_ID, grid 401, is sequenced
1st)
!                             2         201       2                  2 ( "     "  2nd   "   "      "         "  201  "
"         2nd)
!                             3         301       4                  5 ( "     "  5th   "   "      "         "  501  "
"         3rd)
!                             4         401       1                  3 ( "     "  3rd   "   "      "         "  301  "
"         4th)
!                             5         501       3                  1 ( "     "  1st   "   "      "         "  101  "
"         5th)

! NOTE: ARRAYS GRID AND GRID_ID MUST BE SORTED THE SAME AFTER SUBR
!         GRID_PROC HAS RUN

! GRID_ELEM_CONN_ARRAY has NGRID rows, one for each grid i (in numerical order).
! It has a number of cols = 2 + number of elems connected to grid i. A typical array is:

!                   Table of elements connected to each grid

!         Grid      Num elems           ID's of elements connected to this grid -->

!         1011               3          11        1121      1141
!         1012               2          11        12
!         1013               3          12        1323      3143
!         1021               5          11        21        22        1121      2131
```

```
!
**********************************************************************************************************************************
*
! Grid data, con't
! ----------------
```

Adding Element

LINK1/ESP

! Element stiffness processor

! ESP generates the G-set stiffness matrix and puts it into the 1D array STF of nonzero stiffness terms above the diagonal.

! ESP processes the elements sequentially to generate element KE matrix using the EMG set of routines. The element stiffness are transformed from local to basic to global coords for each grid and then merged into the system stiffness, STF, array. See explanation, with an example, in module STF_ARRAYS

Processes all elements on an elmt loop. Within the loop.it calls EMG which generates the element stiffness in local coords

After EMG, it calls ELEM_TRANSFORM_LBG

For each element in calls

Processing flags are set here:

! Set up the option flags for EMG:

```
        OPT(1) = 'N'                        ! OPT(1) is for calc of ME
        OPT(2) = 'N'                        ! OPT(2) is for calc of PTE
        OPT(3) = 'N'                        ! OPT(3) is for calc of SEi, STEi
        OPT(5) = 'N'                        ! OPT(5) is for calc of PPE

        IF      ((SOL_NAME(1:8) == 'BUCKLING') .AND. (LOAD_ISTEP == 2)) THEN
        OPT(4) = 'N'                        ! OPT(4) is for calc of KE-linear
        OPT(6) = 'Y'                        ! OPT(6) is for calc of KE-nonlinear
        ELSE IF ((SOL_NAME(1:8) == 'DIFFEREN') .OR. (SOL_NAME(1:8) == 'NLSTATIC'))
THEN
        OPT(4) = 'Y'                        ! OPT(4) is for calc of KE-linear
        OPT(6) = 'Y'                        ! OPT(6) is for calc of KE-nonlinear
        ELSE
```

```
    OPT(4) = 'Y'                    ! OPT(4) is for calc of KE-linear
    OPT(6) = 'N'                    ! OPT(6) is for calc of KE-nonlinear
    ENDIF
```

LINK1/ESP/EMG

Main driver routine for calculation of matrices for all elements. This routine initializes appropriate arrays and calls other routines to calculate element matrices:

```
!  1) ME         = element mass matrix                    , if OPT(1) = 'Y'
!  2) PTE        = element thermal load vectors           , if OPT(2) = 'Y'
!  3) SEi, STEi, BEi = element stress and strain data recovery matrices, if OPT(3) = 'Y'
!  4) KE         = element linea stiffness matrix         , if OPT(4) = 'Y'
!  5) PPE        = element pressure load matrix           , if OPT(5) = 'Y'
!  6) KED        = element differen stiff matrix calc     , if OPT(6) = 'Y'

! Also, calculate:

!  TE    = Coord transformation matrix (basic to elem)
!  ZS    = Stress recovery coeff's
!  FCONV        = Constants to convert stress to engineering force
!         NOTE: may need to calc KE to ge
```

Called for each element to call element matrices. There are two options that control the specific type: min4 and min4t which is set by parameter,quad4typ whole default is min4t. This generates the plate behavior using4 overlapping tria elements. There is a quad4k option in QDEL1, which is called by EMG if type = quad4: This option was not coded. Can we use this?

I prefer to use param,quad4typ=comlab. Let’s see if ths works.

ELMDAT1

! Generates small arrays of elem data, for use by subroutine EMG, one elem at a time for all elems. Arrays generated are:

```
!   XEB          : basic coords of grids for 1 elem
!   V VECTOR : for some 1-D elems
!   EPROP       : array of elem geometric properties
!   ISOLID   : data for 3-D elems defining options from the PSOLID Bulk Data entry
!   EMAT         : material property data
!   PIN FLAG : Pin flag data for some elems
!   OFFSETS  : offsets for some elem
```

ELMGM2

Called by EMG. Calcs and checks elem geometry for quad elems and provides a transformation matrix (TE) to transfer the elem stiffness matrix! in the elem system to the basic coordinate system.

Calculates grid point coords in local coord system.

To define the elem coordinate system, a mean plane is defined which lies midway between the grid points (HBAR is mean dist).

The elem z direction is in the direction of the cross product of the diagonals (V13 x V24).Initially, the x axis is along side 1-2 of the elem projection onto the mean plane.

For elems thet are not rectangular, the x,y axes are rotated such that x splits the angle between the diagonals.

For each element type, such as ‘quad4’, ELDAT2 is called: (Generates small arrays of elem data, for use by subroutine EMG, one elem at a time for all elems. Arrays generated are:
DT (1 elem temperatures) and PRESS (1 element pressure load))

If type is quad4 then QDEL1 Is called

QDEL1

! Calculates, or calls subr's to calculate, quadrilateral element matrices:

```
!  1) ME    = element mass matrix               , if OPT(1) = 'Y'
!  2) PTE   = element thermal load vectors       , if OPT(2) = 'Y'
!  3) SEi, STEi = element stress data recovery matrices, if OPT(3) = 'Y'
!  4) KE    = element linea stiffness matrix     , if OPT(4) = 'Y'
!  5) PPE   = element pressure load matrix       , if OPT(5) = 'Y'
!  6) KED   = element differen stiff matrix calc   , if OPT(6) = 'Y' = 'Y'

! Modified 10/17/2022 by Harry Schaeffer: Add PARAM,quad4typ,newq4 it COMLAB
quad4 element, quad4rev1, is used to calculate the element stiffness. NB this
PARAM is used at line 216 to calculate the MIN4 type if quad4typ is the
default.
```

The Elements are processed in the **elem**-loop (line 171) to create the KGG-matrix

EMG is called for each element

Each element matrix, ke, is processed in two loops: the outer loop is labeled kgg_rows; and, the inner loop ia labeled kgg_cols: it is generated in LOCAL coords and then transformed into basic coords(? check this, Bill says the transformation is from local (L) to basic(B) and the to global (G). The result is called BIG_KE (check this).

Two params are used: SPARSTOR and EPSIL

Each non-zero term ot the element matrix, it is added to KGG using a linked list that is created in the **stfpnt0**-loop

To add a new element:

Element routines for NE (new element) must be added to EMG. What is not clear is how the infrastructure must change. Replacement might be easier. Let's use the type=quad4 for the comlab shell and replace tne associated stiffness routines with "quad4_comlab"

ELEM_TRANSFORM_LBG

Transforms one element stiff, mass, thermal load or pressure load matrix from local to basic to global coords at each grid including the effects of element offsets. The element matrix is input in array ZE or QE in local element coords.

The output is array ZE or QE containing the element stiff or mass matrix in global coords at each grid. Note that plate element offsets were processed in subr EMG since those offsets are in local element coords, while the BAR and BEAM offsets are handled here after transforming their matrices from local-basic-global, since BAR and BEAM offsets are specified (in the input data) in global coordinates and the BUSH in a unique system

HGS: Don't agree that the stiffness should be Global. NASTRAN transforms to BASIC

.